

pgModeler: muito mais que um modelador de bancos de dados PostgreSQL

Raphael Araújo e Silva

- Bacharel em Ciência da Computação pela Universidade Federal do Tocantins;
- Há 12 anos atuando como Desenvolvedor de Software na Assembleia Legislativa do Estado do Tocantins;
- Apaixonado pela programação, teoria de compiladores, computação gráfica, processamento de imagens e entusiasta da área de banco de dados.

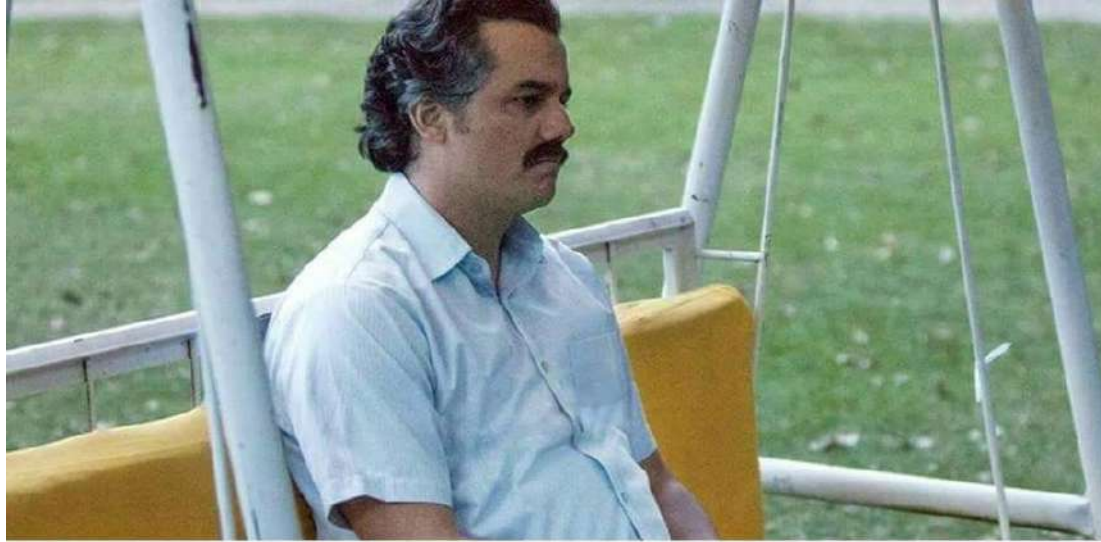


raphael@pgmodeler.io

Como tudo começou...

- Necessidade de documentação para os sistemas que desenvolvia;
- Escassez de ferramentas livres ou baratas para modelagem de dados e geração de código;
- As poucas ferramentas que encontrava à época não atendiam por completo às demandas.

#chateado :(



Hello, world!

- Em 2006 nascia o projeto PostgreSQL Database Modeler ou **pgsqlDBM**. Anos depois alterado para **pgModeler**;
- Concebido sem intenção de divulgação, porém, em 2012 foi liberado sob licença **GPL2** e posteriormente atualizado para **GPL3**.



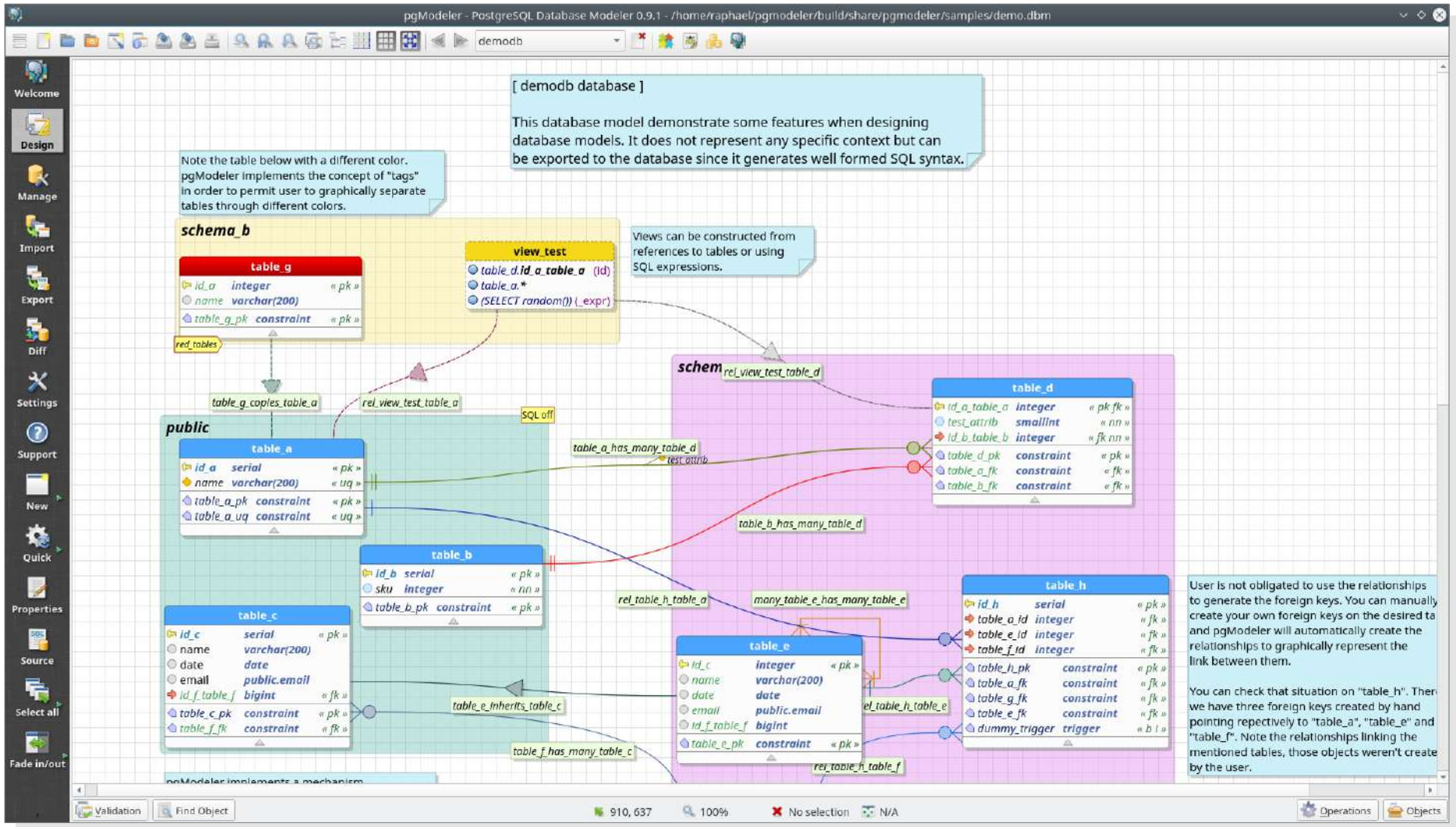
Hello, world!

pgModeler 0.1.2 (Agosto, 2012)

The screenshot displays the pgModeler 0.1.2 application window. The main workspace shows a database model with three tables: 'public: tableA', 'rel: tableB_tableA', and 'public: tableB'. 'tableA' has a primary key 'id' of type 'smallint' and a foreign key 'id' of type 'integer' that references 'tableB'. 'tableB' has a primary key 'id' of type 'serial'. A text box labeled 'This is a text box' is also present in the model. A dialog box titled 'Sobre o pgModeler' (About pgModeler) is overlaid on the workspace, providing information about the software, including its version (0.1.2), license (GNU GPL), and contact information. The dialog box also lists the names of the developers: Michel de Almeida, Filipe Santana, Jonas Nepomuceno, Ricardo Ishibashi, and Alvaro Nunes. The right sidebar shows a tree view of the model objects, including 'new database', 'Caixa de Texto (1)', 'Conversão de Tipo (0)', 'Espaço de Tabela (0)', 'Esquema (1)', 'Linguagem (3)', 'Papel (0)', 'Relacionamento: Tabela-Tabela (1)', and 'Relacionamento: Tabela-Visão (0)'. The bottom status bar indicates 'Operações: 4' and 'Posição: 4'.

Hello, world!

pgModeler 0.9.1 (Maio, 2018)



Do que ele é feito?



- Implementado em sua maior parte em **ISO C++03** (migrando para **ISO C++11/14**) e utilizando o framework **Qt**;
- Compatível com as 3 principais plataformas desktop: **Linux**, **Windows** e **macOS**;
- Encapsula funções da **libpq** para operações em banco de dados e da **libxml2** para geração de diversos tipos de arquivos;
- Implementa uma micro linguagem de templating para geração de código SQL e XML de forma dinâmica.

Do que ele é capaz?

- Modelagem de dados utilizando conceitos clássicos de entidade–relacionamento;
- Geração de código SQL: modele uma vez e exporte para diferentes versões do PostgreSQL (9.x e 10.x);
- Validação da modelagem de dados com aplicação de correções semi–automáticas;
- Engenharia reversa de BD consultando os catálogos do sistema: **pg_catalog.*** e **information_schema.***;

Do que ele é capaz?

- Comparação de banco de dados e geração de código SQL para sincronização de entidades;
- Gerenciamento de banco de dados e manipulação dos dados de tabelas de forma facilitada;
- Possui uma versão CLI (linha de comando) que encapsula algumas das principais funcionalidades;
- Extras: suporte a tipos de dados geoespaciais (PostGIS), suporte a plug-ins, mecanismo de segurança para resguardar o trabalho em caso de bugs ou crashes, e muitos outros...

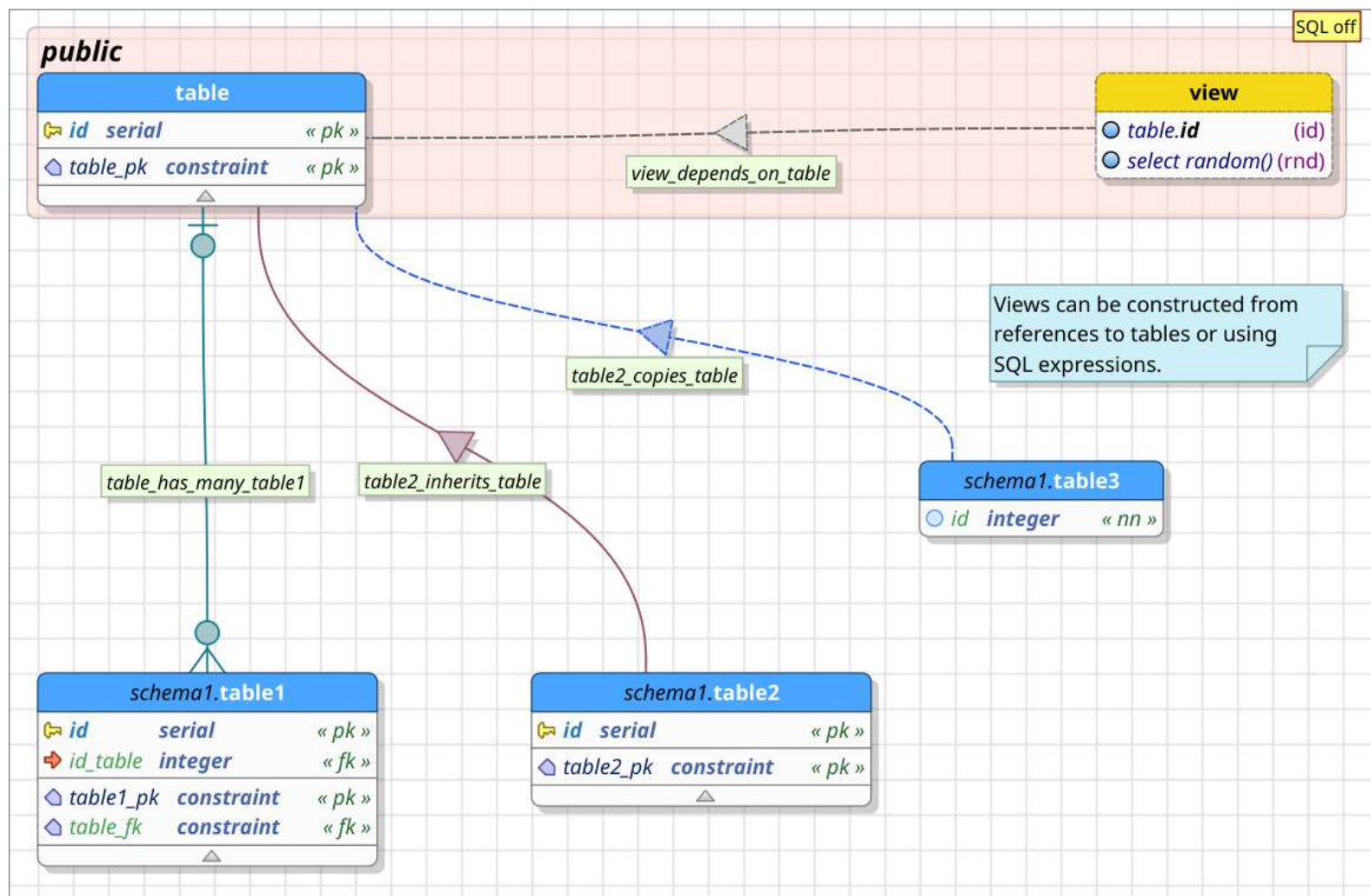
Modelagem de dados

Modelagem de dados

- Documentação, visão macro de um banco de dados, geração de código;
- Permite o agrupamento visual de objetos por esquemas ou tags;
- Modelos podem ser exportados para PNG ou SVG;
- Implementa um mecanismo de propagação automática de colunas e restrições;

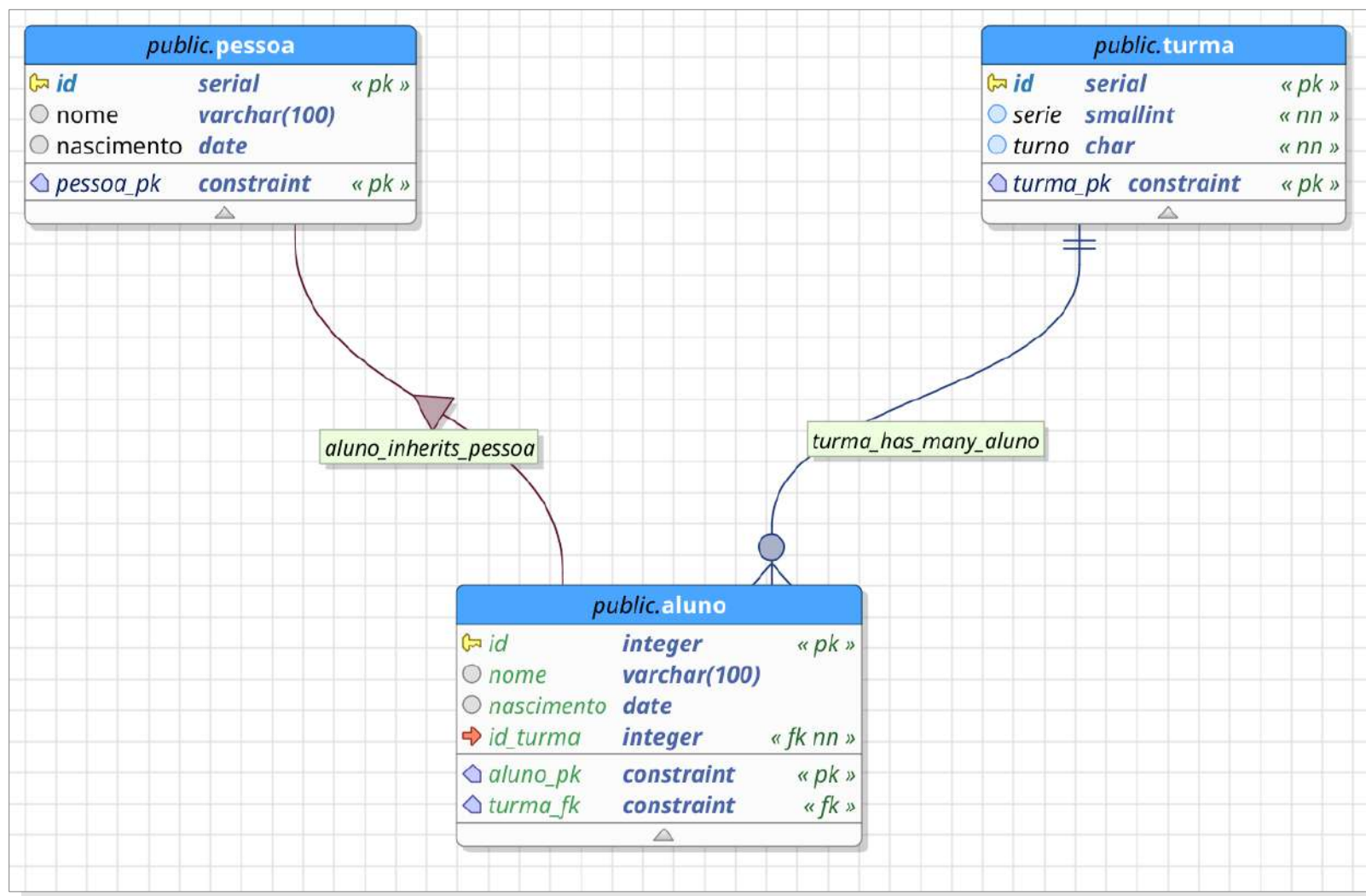
Modelagem de dados

- Entidades gráficas:



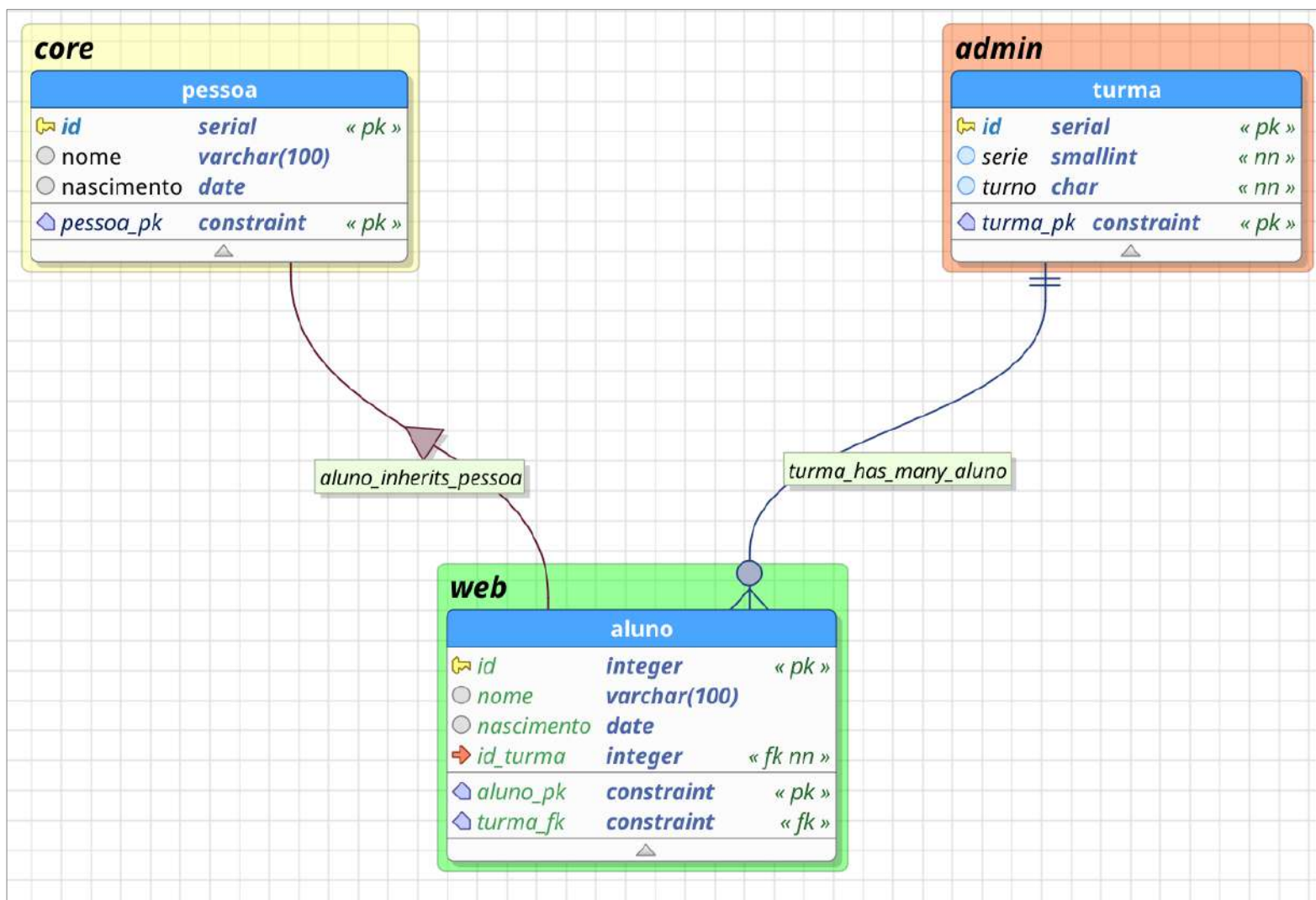
Modelagem de dados

- Propagação de colunas e restrições:



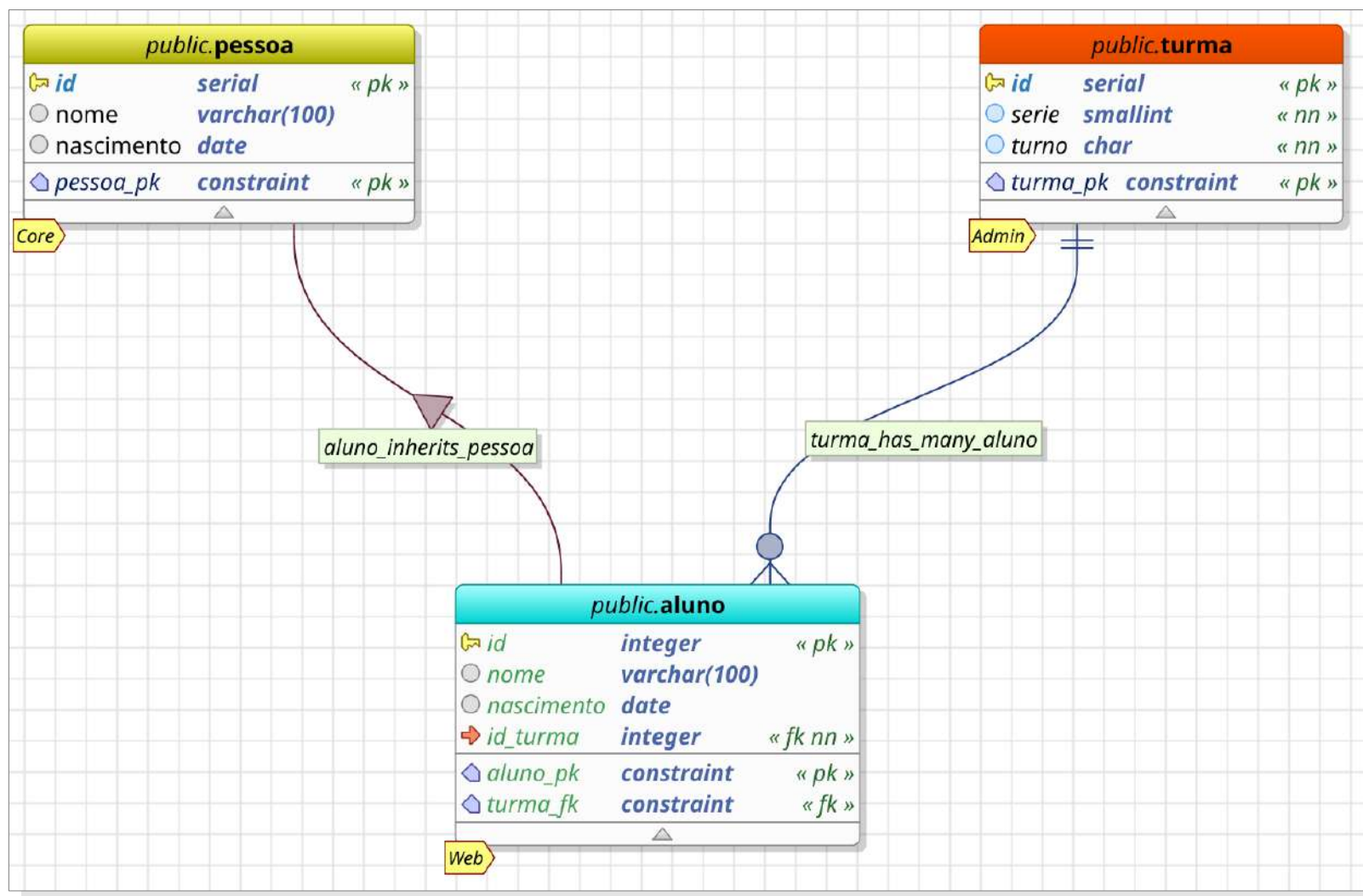
Modelagem de dados

- Agrupamento por esquemas:



Modelagem de dados

- Agrupamento por tags:



Geração de código SQL

Geração de código SQL

- É a função primordial da ferramenta;
- Ocorre em 3 etapas:
 1. Extração dos atributos dos objetos;
 2. Parsing dos templates dos objetos;
 3. Conversão dos templates interpretados em código SQL;

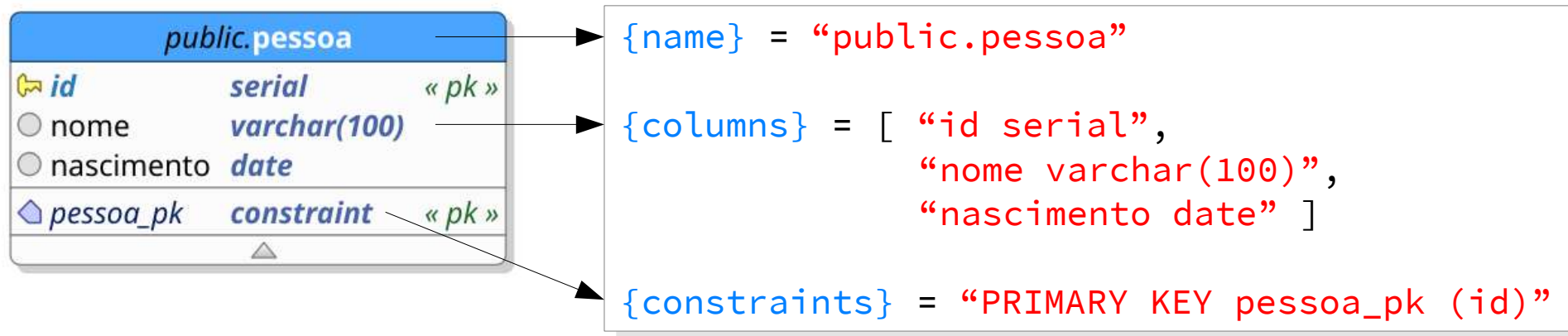
Geração de código SQL

1. Extração dos atributos do objeto:

<i>public.pessoa</i>		
 <i>id</i>	<i>serial</i>	« pk »
<input type="radio"/> nome	<i>varchar(100)</i>	
<input type="radio"/> nascimento	<i>date</i>	
 <i>pessoa_pk</i>	<i>constraint</i>	« pk »

Geração de código SQL

1. Extração dos atributos do objeto:



Geração de código SQL

2. Parsing do template do objeto:

```
{name} = "public.pessoa"  
  
{columns} = [ "id serial",  
              "nome varchar(100)",  
              "nascimento date" ]  
  
{constraints} = "PRIMARY KEY  
                pessoa_pk (id)"
```

Geração de código SQL

2. Parsing do template do objeto:

```
{name} = "public.pessoa"

{columns} = [ "id serial",
              "nome varchar(100)",
              "nascimento date" ]

{constraints} = "PRIMARY KEY
                pessoa_pk (id)"
```

```
# SQL definition for tables
[-- object: ] {name} [ | type: ] {sql-object} [ --] $br
[-- ] {drop}

[CREATE]

%if ({pgsql-ver} != "9.0") %and {unlogged} %then
    [ UNLOGGED]
%end

[ TABLE ] {name} ( $br

%if {columns} %then
    {columns}
%end

%if {constraints} %then
    {constraints}
%end

$br );

[-- ddl-end --] $br
```

Geração de código SQL

3. Código SQL resultante:

```
# SQL definition for tables
[-- object: ] {name} [ | type: ] {sql-object} [ --] $br
[-- ] {drop}

[CREATE]

%if ({pgsql-ver} != "9.0") %and {unlogged} %then
    [ UNLOGGED]
%end

[ TABLE ] {name} ( $br

%if {columns} %then
    {columns}
%end

%if {constraints} %then
    {constraints}
%end

$br );
[-- ddl-end --] $br
```

```
-- object: public.pessoa | type: TABLE --
-- DROP TABLE IF EXISTS public.pessoa CASCADE;
CREATE TABLE public.pessoa(
    id serial NOT NULL,
    nome varchar(100),
    nascimento date,
    CONSTRAINT pessoa_pk PRIMARY KEY (id)
);
-- ddl-end --
```

Geração de código SQL

The image shows a screenshot of the pgModeler application. On the left, a table definition for `public.pessoa` is displayed. The table has three columns: `id` (type `serial`, primary key), `nome` (type `varchar(100)`), and `nascimento` (type `date`). A primary key constraint named `pessoa_pk` is defined on the `id` column.

On the right, a "Source code visualization" window is open, showing the SQL code generated for the table. The code is as follows:

```
1 -- object: public.pessoa | type: TABLE --
2 -- DROP TABLE IF EXISTS public.pessoa CASCADE;
3 CREATE TABLE public.pessoa(
4     id serial NOT NULL,
5     nome varchar(100),
6     nascimento date,
7     CONSTRAINT pessoa_pk PRIMARY KEY (id)
8
9 );
10 -- ddl-end --
11 ALTER TABLE public.pessoa OWNER TO postgres;
12 -- ddl-end --
13
14
```

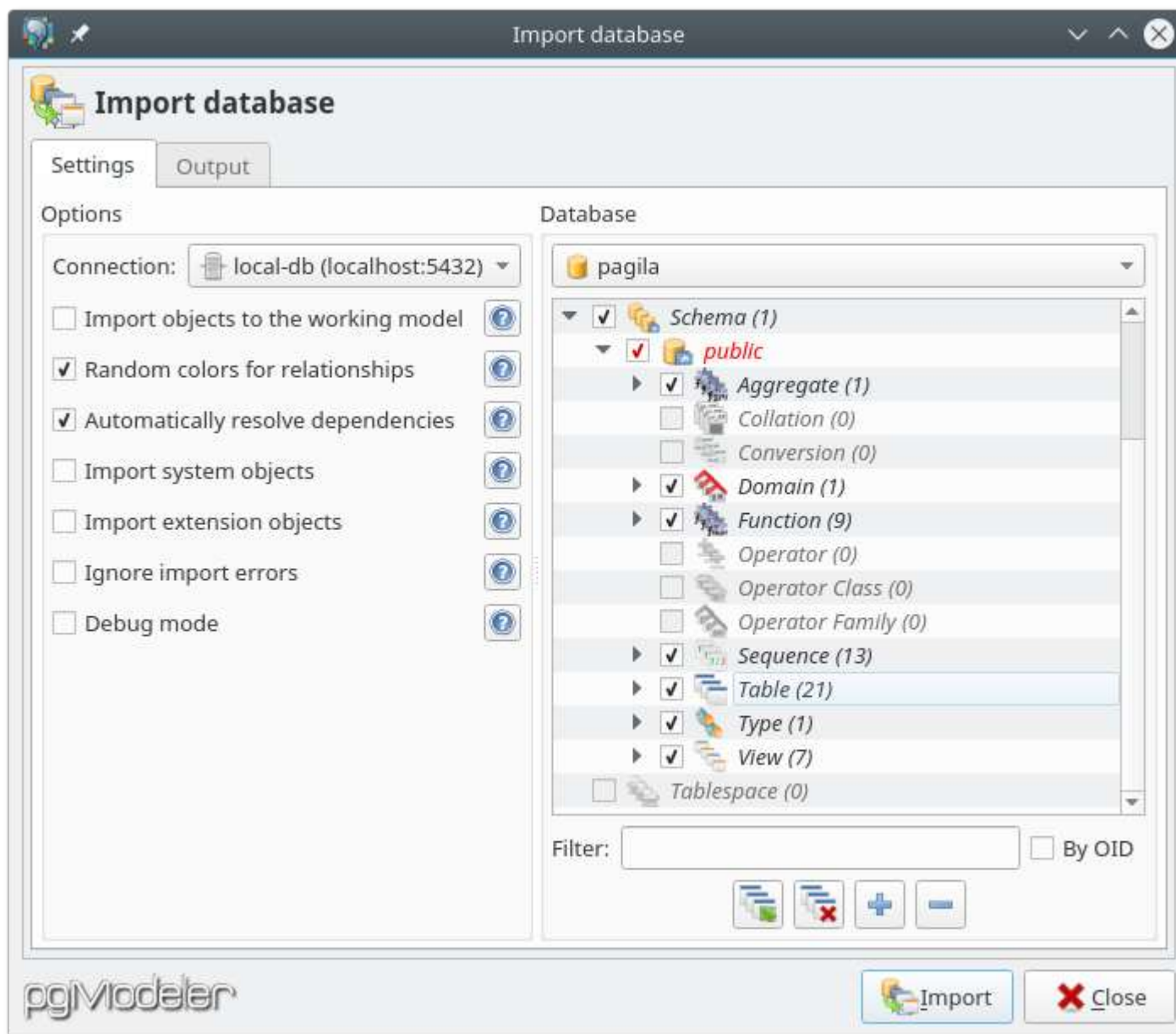
The "Version" dropdown menu in the window is set to "PostgreSQL 10.0". The "Code display" dropdown is set to "Original". There are "Save SQL" and "Ok" buttons at the bottom of the window.

Engenharia reversa

Engenharia reversa

- Leitura dos catálogos do sistema para a reconstrução do modelo de banco de dados;
- Permite a importação de um banco de dados (ou frações deste) resolvendo dependências automaticamente;
- Executada em três etapas:
 1. Parsing do template e geração da query de catálogo;
 2. Geração de um código XML que representa o objeto do banco de dados;
 3. Interpretação do código XML criando o objeto no modelo.

Engenharia reversa



Engenharia reversa

1. Parsing do template e geração da query de catálogo:

```
[ SELECT tb.oid, tb.relname AS name, tb.relnamespace AS schema,  
tb.relowner AS owner, tb.reltuples::int8 AS row_amount,  
tb.reltablespace AS tablespace, tb.relacl AS permission,  
relhasoids AS oids_bool, ]  
  
%if ({pgsql-ver} == "9.0") %then  
    [ FALSE AS unlogged_bool, ]  
%else  
    [ CASE relpersistence  
      WHEN 'u' THEN TRUE  
      ELSE FALSE  
      END  
      AS unlogged_bool, ]  
%end  
  
%if ({pgsql-ver} <f "9.5") %then  
    [ FALSE AS rls_enabled_bool, ]  
    [ FALSE AS rls_forced_bool, ]  
%else  
    [ tb.relrowsecurity AS rls_enabled_bool, ]  
    [ tb.relforcerowsecurity AS rls_forced_bool, ]  
%end  
  
({comment}) [ AS comment ]  
  
[ FROM pg_class AS tb  
  LEFT JOIN pg_tables AS _tb1 ON _tb1.tablename=tb.relname  
  WHERE tb.relkind='r' ]  
  
%if {schema} %then [ AND _tb1.schemaname= ] '{schema}' %end
```

Engenharia reversa

1. Parsing do template e geração da query de catálogo:

```
SELECT tb.oid, tb.relname AS name, tb.relnamespace AS schema,
       tb.relowner AS owner, tb.reltuples::int8 AS row_amount,
       tb.reltablespace AS tablespace, tb.relacl AS permission,
       relhasoids AS oids_bool,
       CASE relpersistence WHEN 'u' THEN TRUE
       ELSE FALSE END AS unlogged_bool,
       tb.relrowsecurity AS rls_enabled_bool, tb.relforcerowsecurity AS rls_forced_bool,
       (SELECT array_agg(inhparent) AS parents FROM pg_inherits WHERE inhrelid = tb.oid),
       (SELECT description FROM pg_description WHERE objoid = tb.oid AND objsubid = 0) AS comment
FROM pg_class AS tb
LEFT JOIN pg_tables AS _tb1 ON _tb1.tablename=tb.relname
WHERE tb.relkind='r' AND
       tb.oid > 13018 AND
       (tb.oid NOT IN (13006,13007,13008,13009)) AND
       tb.oid IN (26551,26557,26565);
```

Engenharia reversa

2. Geração do código XML que representa o objeto:

```
<table name="pessoa">
  <schema name="public"/>
  <role name="postgres"/>
  <position x="500" y="200"/>
  <column name="id" not-null="true">
    <type name="serial" length="0"/>
  </column>
  <column name="nome">
    <type name="varchar" length="100"/>
  </column>
  <column name="nascimento">
    <type name="date" length="0"/>
  </column>
  <constraint name="pessoa_pk" type="pk-constr" table="public.pessoa">
    <columns names="id" ref-type="src-columns"/>
  </constraint>
</table>
```

Engenharia reversa

3. Geração do objeto a partir do código XML:

```
<table name="pessoa">
  <schema name="public"/>
  <role name="postgres"/>
  <position x="500" y="200"/>
  <column name="id" not-null="true">
    <type name="serial" length="0"/>
  </column>
  <column name="nome">
    <type name="varchar" length="100"/>
  </column>
  <column name="nascimento">
    <type name="date" length="0"/>
  </column>
  <constraint name="pessoa_pk" type="pk-constr" table="public.pessoa">
    <columns names="id" ref-type="src-columns"/>
  </constraint>
</table>
```

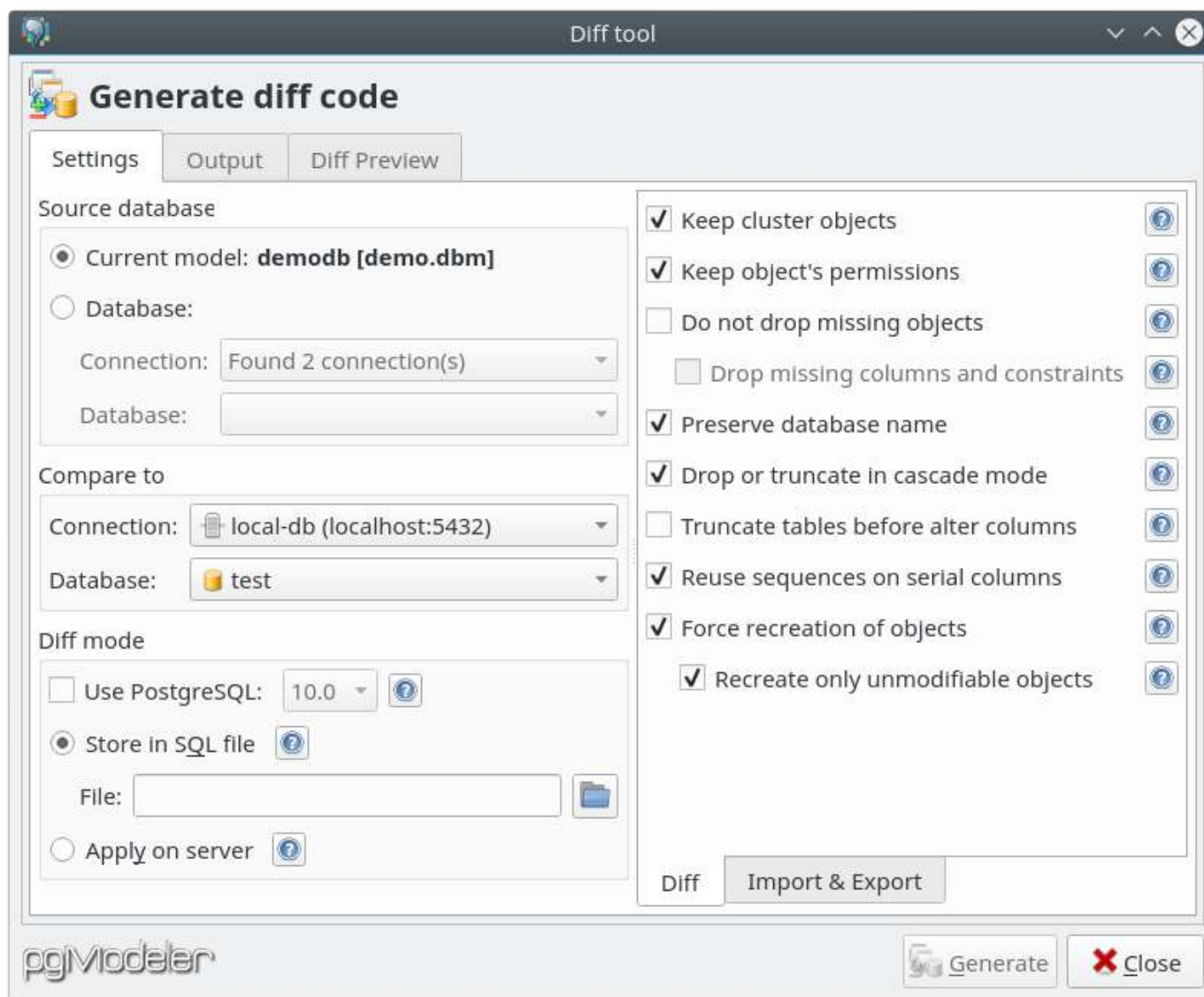
public.pessoa		
 id	serial	« pk »
<input type="radio"/> nome	varchar(100)	
<input type="radio"/> nascimento	date	
 pessoa_pk	constraint	« pk »

Comparação de BDs (diff)

Comparação de banco de dados

- Detecta diferenças entre um modelo e um banco de dados ou entre dois bancos de dados;
- Gera o código SQL capaz de sincronizar o banco de dados deixando-o semelhante ao modelo ou BD utilizado como entrada;

Comparação de banco de dados



Comparação de banco de dados

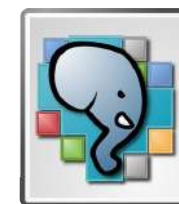
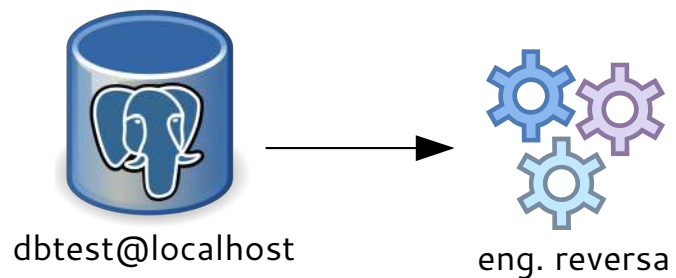


dbtest@localhost



dbtest.dbm

Comparação de banco de dados

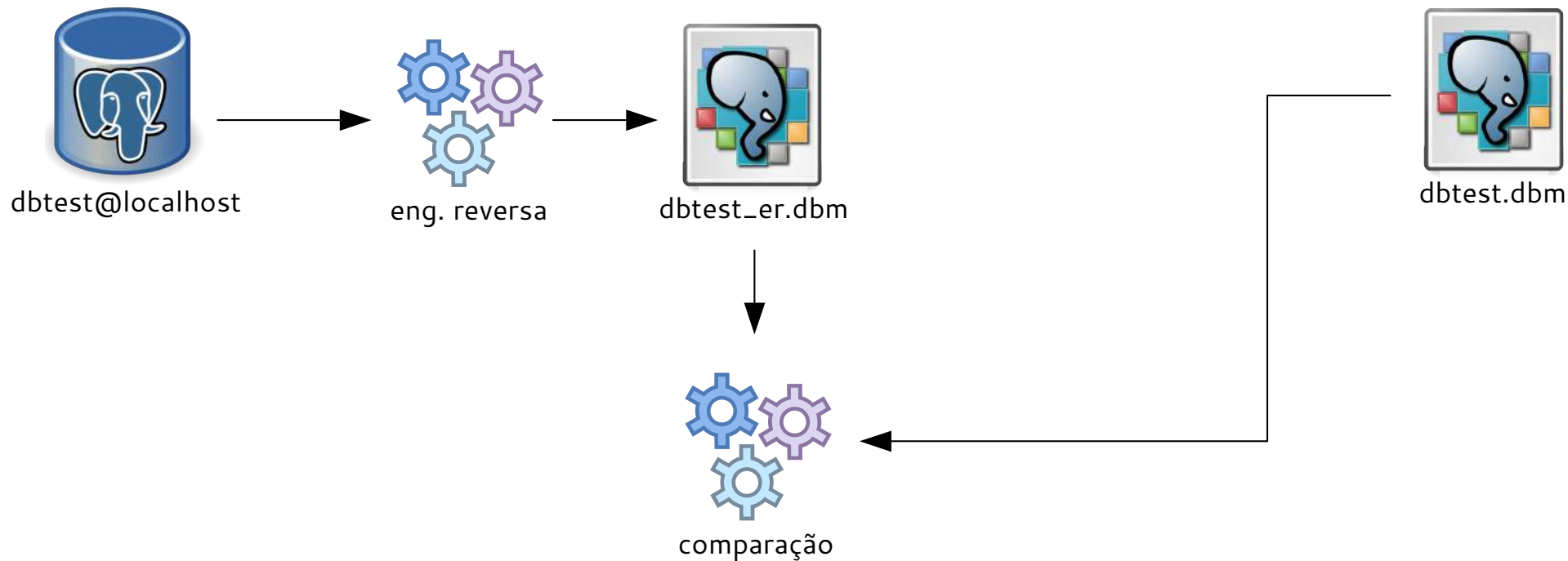


dbtest.dbm

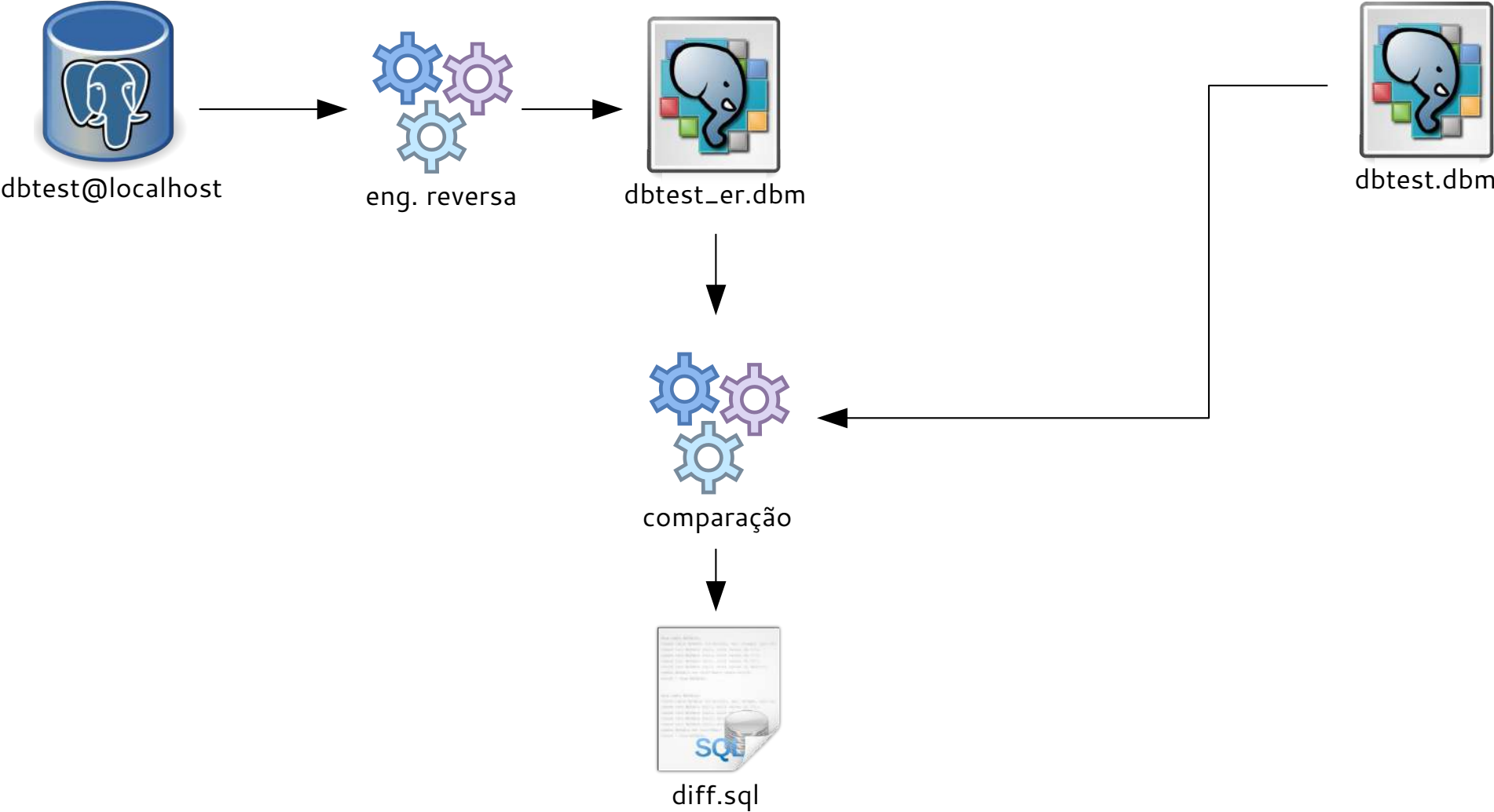
Comparação de banco de dados



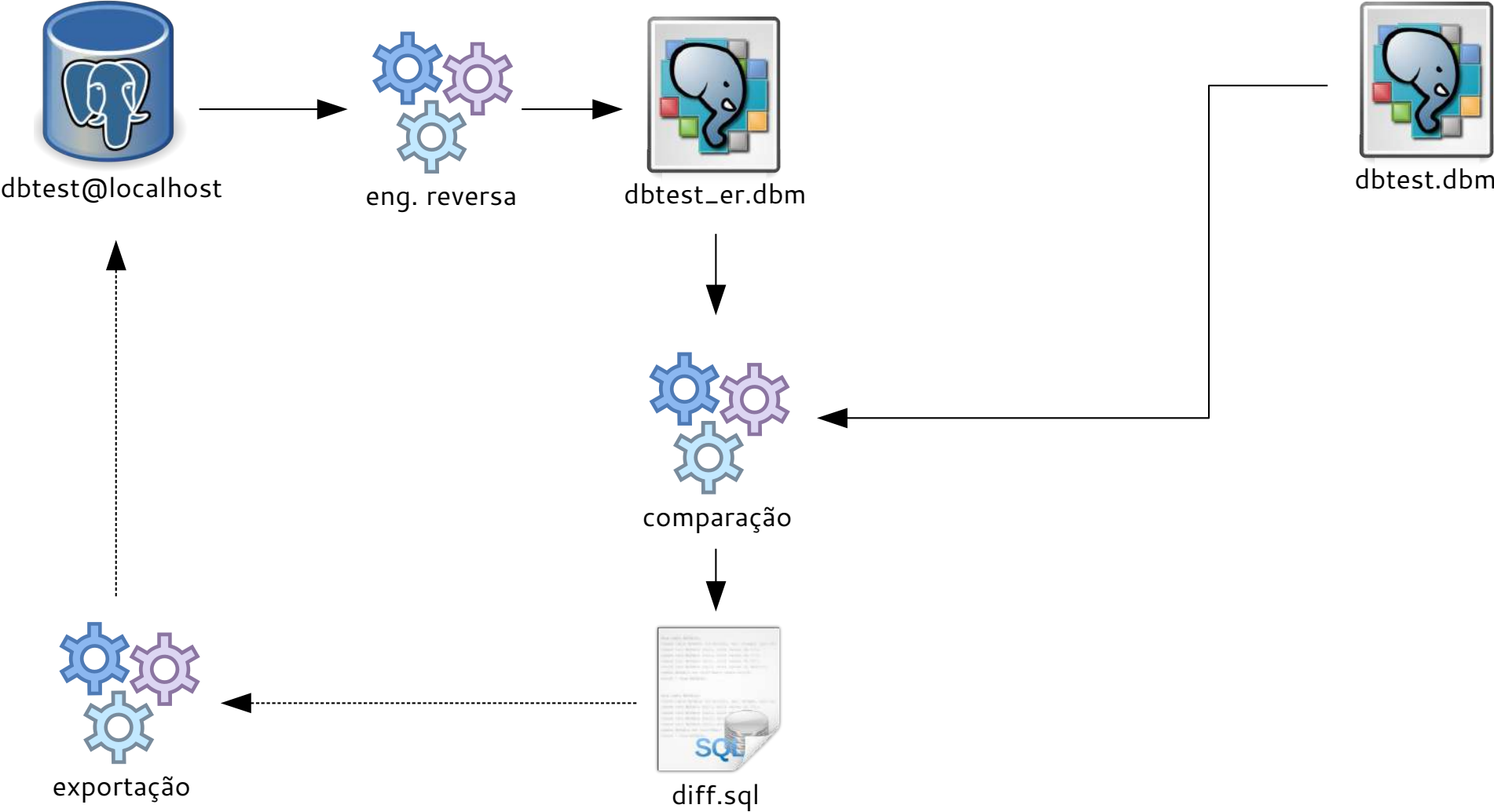
Comparação de banco de dados



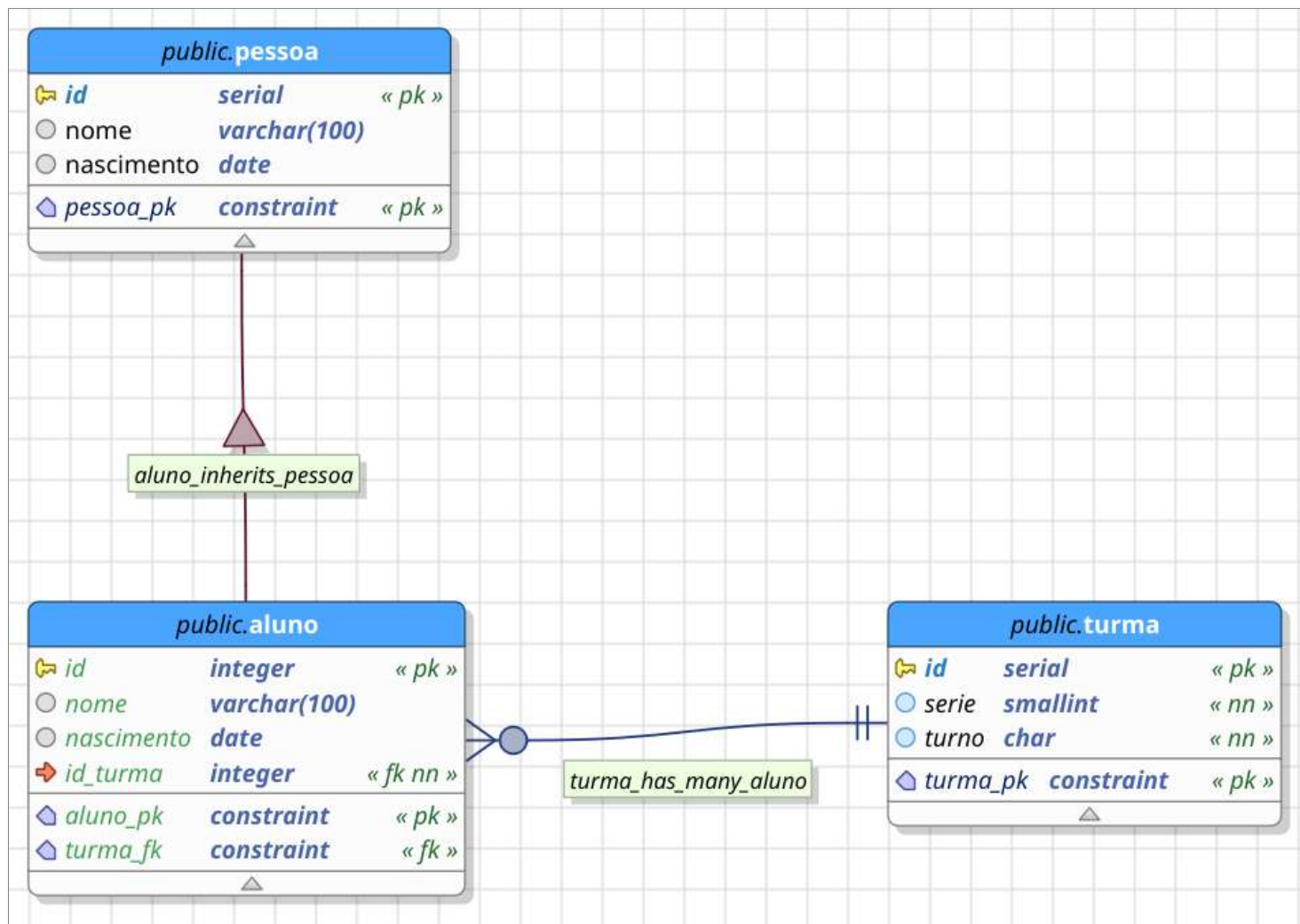
Comparação de banco de dados



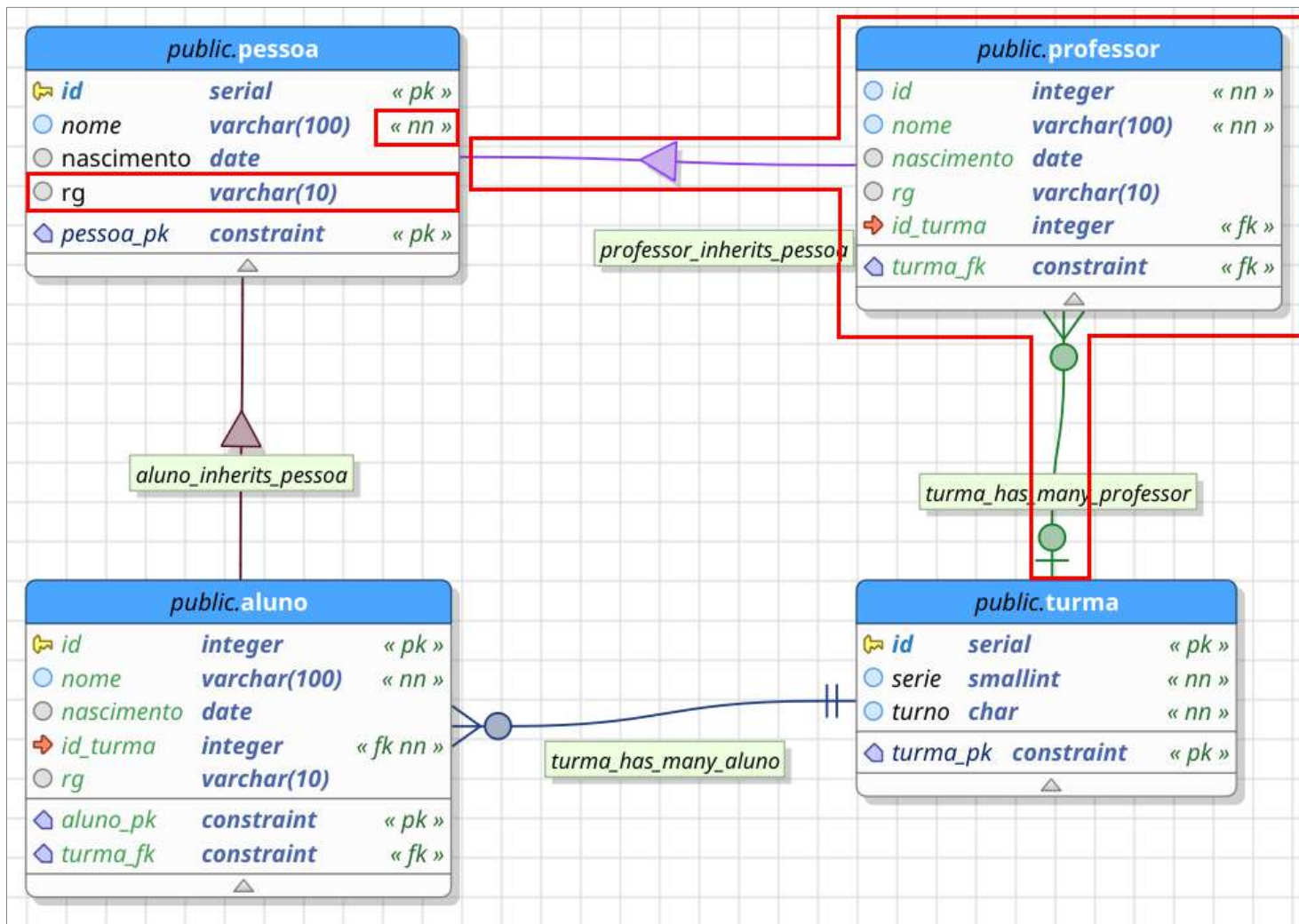
Comparação de banco de dados



Comparação de banco de dados



Comparação de banco de dados



Comparação de banco de dados

```
1 SET search_path=public,pg_catalog,core,web,admin;
2 -- ddl-end --
3
4 -- [ Created objects ] --
5 -- object: rg | type: COLUMN --
6 -- ALTER TABLE public.pessoa DROP COLUMN IF EXISTS rg CASCADE;
7 ALTER TABLE public.pessoa ADD COLUMN rg varchar(10);
8 -- ddl-end --
9
10 -- object: public.professor | type: TABLE --
11 -- DROP TABLE IF EXISTS public.professor CASCADE;
12 CREATE TABLE public.professor(
13     id_turma integer
14 -- id integer NOT NULL,
15 -- nome varchar(100) NOT NULL,
16 -- nascimento date,
17 -- rg varchar(10),
18
19 ) INHERITS(public.pessoa)
20 ;
21 -- ddl-end --
22 ALTER TABLE public.professor OWNER TO postgres;
23 -- ddl-end --
24
25 -- [ Changed objects ] --
26 ALTER TABLE public.pessoa ALTER COLUMN nome SET NOT NULL;
27 -- ddl-end --
28
29 -- [ Created foreign keys ] --
30 -- object: turma_fk | type: CONSTRAINT --
31 -- ALTER TABLE public.professor DROP CONSTRAINT IF EXISTS turma_fk CASCADE;
32 ALTER TABLE public.professor ADD CONSTRAINT turma_fk FOREIGN KEY (id_turma)
33 REFERENCES public.turma (id) MATCH FULL
34 ON DELETE SET NULL ON UPDATE CASCADE;
35 -- ddl-end --
```

Administração de BDs

Administração de banco de dados

The screenshot displays the pgModeler PostgreSQL Database Modeler 0.9.1 interface. The left sidebar shows a tree view of the database structure, including schemas (admin, core, public) and tables (aluno, pessoa, turma). The main window shows a SQL query in the 'SQL execution' pane, which is highlighted in yellow. Below the query, a table of results is displayed, showing columns for oid, name, schema, owner, row_amount, tablespace, permission, oids_bool, unlogged_bool, rls_enabled_bool, rls_forced_bool, parents, and comment. The results table contains three rows of data. At the bottom, the 'Source code' pane shows the SQL script used to create the table.

```
1 SELECT tb.oid, tb.relname AS name, tb.relnamespace AS schema,  
2        tb.relowner AS owner, tb.reltuples::int8 AS row_amount,  
3        tb.reltablespace AS tablespace, tb.relacl AS permission,  
4        relhasoids AS oids_bool,  
5        CASE relpersistence WHEN 'u' THEN TRUE  
6        ELSE FALSE END AS unlogged_bool,  
7        tb.relrowsecurity AS rls_enabled_bool, tb.relforcerowsecurity AS rls_forced_bool,  
8        (SELECT array_agg(inhparent) AS parents FROM pg_inherits WHERE inhrelid = tb.oid),  
9        (SELECT description FROM pg_description WHERE objoid = tb.oid AND objsubid = 0) AS comment  
10 FROM pg_class AS tb  
11 LEFT JOIN pg_tables AS _tb1 ON _tb1.tablename=tb.relname  
12 WHERE tb.relkind='r' AND  
13        tb.oid > 13018 AND  
14        (tb.oid NOT IN (13006,13007,13008,13009)) AND  
15        tb.oid IN (26551,26557,26565);  
16
```

oid	name	schema	owner	row_amount	tablespace	permission	oids_bool	unlogged_bool	rls_enabled_bool	rls_forced_bool	parents	comment
1 26551	pessoa	2200	10	0	0		f	f	f	f		
2 26557	aluno	2200	10	0	0		f	f	f	f	{26551}	
3 26565	turma	2200	10	0	0		f	f	f	f		

```
1 -- object: public.pessoa | type: TABLE --  
2 -- DROP TABLE IF EXISTS public.pessoa CASCADE;  
3 CREATE TABLE public.pessoa(  
4     id integer NOT NULL DEFAULT nextval('pessoa_id_seq'::regclass),  
5     nome character varying(100),  
6     nascimento date  
7 );  
8 -- ddl-end --  
9 ALTER TABLE public.pessoa OWNER TO postgres;  
10 -- ddl-end --
```

Administração de banco de dados

The screenshot displays the pgModeler PostgreSQL Database Modeler 0.9.1 interface. The left sidebar shows a 'Database explorer' with a tree view of a database named 'new_database'. The tree view is expanded to show a schema named 'public', which contains several objects including 'aluno', 'pessoa', and 'turma'. The 'pessoa' table is selected, and its attributes are listed in a table below the tree view.

Attribute	Value
Comment	
Name	pessoa
Object type	Table
OID	26551
Owner	postgres
Parents	
Permissions	
rls-enabled	
rls-forced	
Rows amount	0

The main window shows the 'SQL execution' panel. The SQL query being executed is:

```
1 SELECT tb.oid, tb.relname AS name, tb.relnamespace AS schema,  
2        tb.relowner AS owner, tb.reltuples::int8 AS row_amount,  
3        tb.reltablespace AS tablespace, tb.relacl AS permission,  
4        relhasoids AS oids_bool,  
5        CASE relpersistence WHEN 'u' THEN TRUE  
6        ELSE FALSE END AS unlogged_bool,  
7        tb.relrowsecurity AS rls_enabled_bool, tb.relforcerowsecurity AS rls_forced_bool,  
8        (SELECT array_agg(inhparent) AS parents FROM pg_inherits WHERE inhrelid = tb.oid),  
9        (SELECT description FROM pg_description WHERE objoid = tb.oid AND objsubid = 0) AS comment  
10 FROM pg_class AS tb  
11 LEFT JOIN pg_tables AS _tb1 ON _tb1.tablename=tb.relname  
12 WHERE tb.relkind='r' AND  
13        tb.oid > 13018 AND  
14        (tb.oid NOT IN (13006,13007,13008,13009)) AND  
15        tb.oid IN (26551,26557,26565);  
16
```

The results of the query are displayed in a table below the SQL execution panel:

oid	name	schema	owner	row_amount	tablespace	permission	oids_bool	unlogged_bool	rls_enabled_bool	rls_forced_bool	parents	comment
1 26551	pessoa	2200	10	0	0		f	f	f	f		
2 26557	aluno	2200	10	0	0		f	f	f	f	{26551}	
3 26565	turma	2200	10	0	0		f	f	f	f		

The bottom panel shows the 'Source code' for the selected table:

```
1 -- object: public.pessoa | type: TABLE --  
2 -- DROP TABLE IF EXISTS public.pessoa CASCADE;  
3 CREATE TABLE public.pessoa(  
4     id integer NOT NULL DEFAULT nextval('pessoa_id_seq'::regclass),  
5     nome character varying(100),  
6     nascimento date  
7 );  
8 -- ddl-end --  
9 ALTER TABLE public.pessoa OWNER TO postgres;  
10 -- ddl-end --
```

Administração de banco de dados

The screenshot displays the pgModeler interface. On the left, the 'Database explorer' shows a tree view of the 'new_database' schema, including tables like 'aluno', 'pessoa', and 'turma'. The main window is titled 'SQL execution' and shows a query being executed against the 'new_database' connection. The query is a SELECT statement that joins 'pg_class' and 'pg_tables' to retrieve table metadata. Below the query, the results are displayed in a table with 12 columns: 'oid', 'name', 'schema', 'owner', 'row_amount', 'tablespace', 'permission', 'oids_bool', 'unlogged_bool', 'rls_enabled_bool', 'rls_forced_bool', 'parents', and 'comment'. Three rows of data are shown, corresponding to tables with oid values 26551, 26557, and 26565. At the bottom, the 'Source code' pane shows the SQL script used to create the 'pessoa' table.

```
SQL execution
new_database X
Script Find Run SQL Clear All Snippets Export Output new_database@localhost:5432

1 SELECT tb.oid, tb.relname AS name, tb.relnamespace AS schema,
2        tb.relowner AS owner, tb.reltuples::int8 AS row_amount,
3        tb.reltablespace AS tablespace, tb.relacl AS permission,
4        relhasoids AS oids_bool,
5        CASE relpersistence WHEN 'u' THEN TRUE
6        ELSE FALSE END AS unlogged_bool,
7        tb.relrowsecurity AS rls_enabled_bool, tb.relforcerowsecurity AS rls_forced_bool,
8        (SELECT array_agg(inhparent) AS parents FROM pg_inherits WHERE inhrelid = tb.oid),
9        (SELECT description FROM pg_description WHERE objoid = tb.oid AND objsubid = 0) AS comment
10 FROM pg_class AS tb
11 LEFT JOIN pg_tables AS _tb1 ON _tb1.tablename=tb.relname
12 WHERE tb.relkind='r' AND
13        tb.oid > 13018 AND
14        (tb.oid NOT IN (13006,13007,13008,13009)) AND
15        tb.oid IN (26551,26557,26565);
16
```

oid	name	schema	owner	row_amount	tablespace	permission	oids_bool	unlogged_bool	rls_enabled_bool	rls_forced_bool	parents	comment
1 26551	pessoa	2200	10	0	0		f	f	f	f		
2 26557	aluno	2200	10	0	0		f	f	f	f	{26551}	
3 26565	turma	2200	10	0	0		f	f	f	f		

```
Results (3) Messages (1) History

Source code
1 -- object: public.pessoa | type: TABLE --
2 -- DROP TABLE IF EXISTS public.pessoa CASCADE;
3 CREATE TABLE public.pessoa(
4     id integer NOT NULL DEFAULT nextval('pessoa_id_seq'::regclass),
5     nome character varying(100),
6     nascimento date
7 );
8 -- ddl-end --
9 ALTER TABLE public.pessoa OWNER TO postgres;
10 -- ddl-end --
```

Administração de banco de dados

The screenshot displays the pgModeler PostgreSQL Database Modeler 0.9.1 interface. The main window is titled "new_database" and shows a SQL execution window with the following query:

```
1 SELECT tb.oid, tb.relname AS name, tb.relnamespace AS schema,  
2        tb.relowner AS owner, tb.reltuples::int8 AS row_amount,  
3        tb.reltablespace AS tablespace, tb.relacl AS permission,  
4        relhasoids AS oids_bool,  
5        CASE relpersistence WHEN 'u' THEN TRUE  
6        ELSE FALSE END AS unlogged_bool,  
7        tb.relrowsecurity AS rls_enabled_bool, tb.relforcerowsecurity AS rls_forced_bool,  
8        (SELECT array_agg(inhparent) AS parents FROM pg_inherits WHERE inhrelid = tb.oid),  
9        (SELECT description FROM pg_description WHERE objoid = tb.oid AND objsubid = 0) AS comment  
10 FROM pg_class AS tb  
11 LEFT JOIN pg_tables AS _tbl ON _tbl.tablename=tb.relname  
12 WHERE tb.relkind='r' AND  
13        tb.oid > 13018 AND  
14        (tb.oid NOT IN (13006,13007,13008,13009)) AND  
15        tb.oid IN (26551,26557,26565);
```

The results of the query are displayed in a table with the following columns: oid, name, schema, owner, row_amount, tablespace, permission, oids_bool, unlogged_bool, rls_enabled_bool, rls_forced_bool, parents, comment.

oid	name	schema	owner	row_amount	tablespace	permission	oids_bool	unlogged_bool	rls_enabled_bool	rls_forced_bool	parents	comment
1 26551	peessoa	2200	10	0	0		f	f	f	f		
2 26557	aluno	2200	10	0	0		f	f	f	f	{26551}	
3 26565	turma	2200	10	0	0		f	f	f	f		

The interface also shows a "Database explorer" on the left with a tree view of the database structure, including schemas (admin, core, public) and tables (aluno, pessoa, turma). The "Source code" window at the bottom shows the following SQL code:

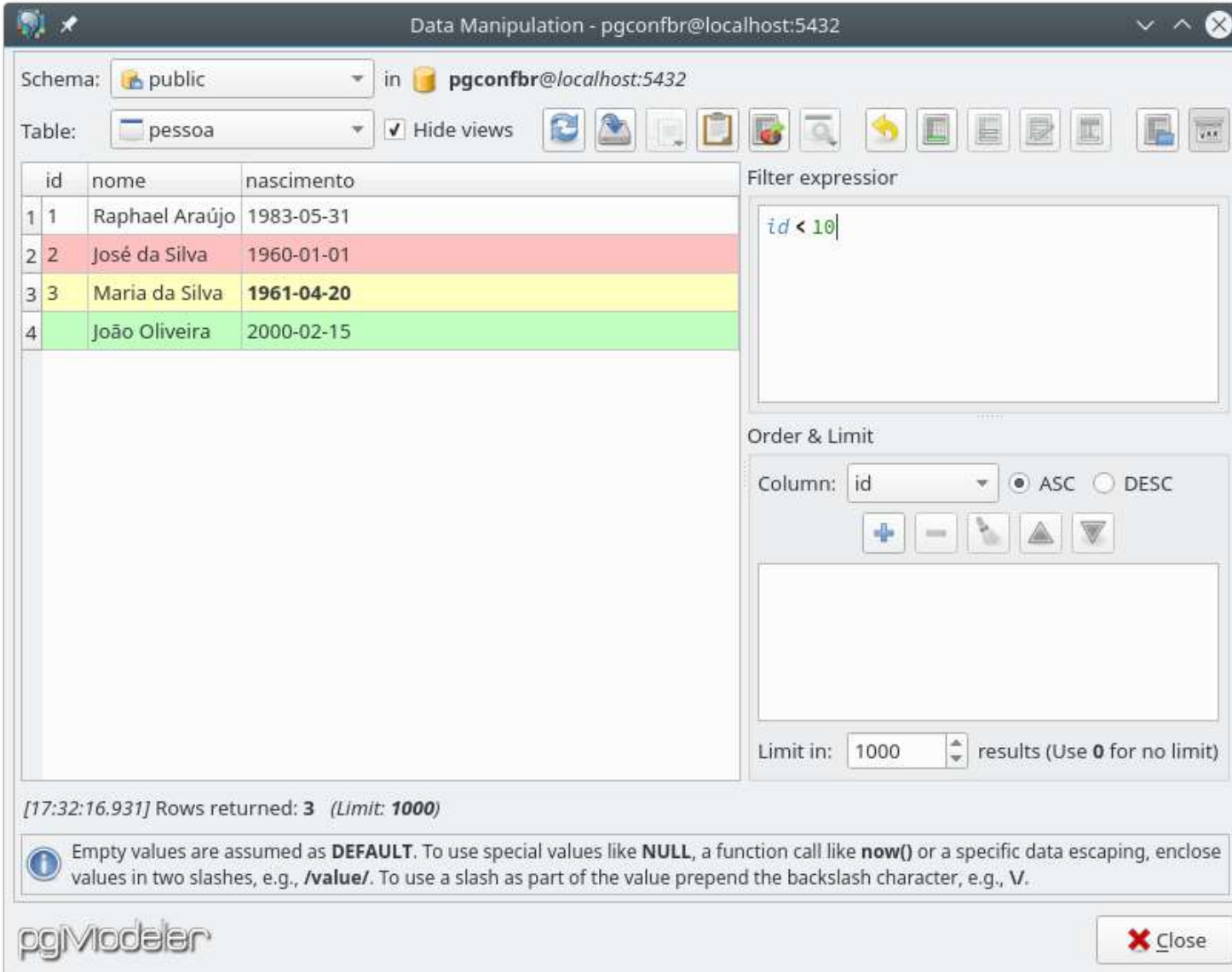
```
1 -- object: public.pessoa | type: TABLE --  
2 -- DROP TABLE IF EXISTS public.pessoa CASCADE;  
3 CREATE TABLE public.pessoa(  
4     id integer NOT NULL DEFAULT nextval('pessoa_id_seq'::regclass),  
5     nome character varying(100),  
6     nascimento date  
7 );  
8 -- ddl-end --  
9 ALTER TABLE public.pessoa OWNER TO postgres;  
10 -- ddl-end --
```


Manipulação de dados

Manipulação de dados

- Feita em formulário dedicado;
- Destaca cada tipo de operação sobre um registro;
- Permite a navegação entre registros que se relacionam;
- Extras: filtragem e ordenação, exportação de resultados para CSV, alimentação de dados via CSV;

Manipulação de dados



The screenshot shows the 'Data Manipulation' window in pgModeler, connected to a PostgreSQL database at localhost:5432. The window title is 'Data Manipulation - pgconfbr@localhost:5432'. The 'Schema' is set to 'public' and the 'Table' is 'pessoa'. The 'Filter expression' is 'id < 10'. The 'Order & Limit' section shows 'Column: id' with 'ASC' selected and a 'Limit in: 1000' results. The table displays 4 rows of data, with the first three rows highlighted in red, yellow, and green respectively. The status bar at the bottom indicates '[17:32:16.931] Rows returned: 3 (Limit: 1000)'. A help message explains that empty values are assumed as DEFAULT and provides instructions for using special values like NULL and escaping slashes.

Schema: public in pgconfbr@localhost:5432

Table: pessoa Hide views

	id	nome	nascimento
1	1	Raphael Araújo	1983-05-31
2	2	José da Silva	1960-01-01
3	3	Maria da Silva	1961-04-20
4	4	João Oliveira	2000-02-15

Filter expression: `id < 10`

Order & Limit: Column: id ASC DESC

Limit in: 1000 results (Use 0 for no limit)

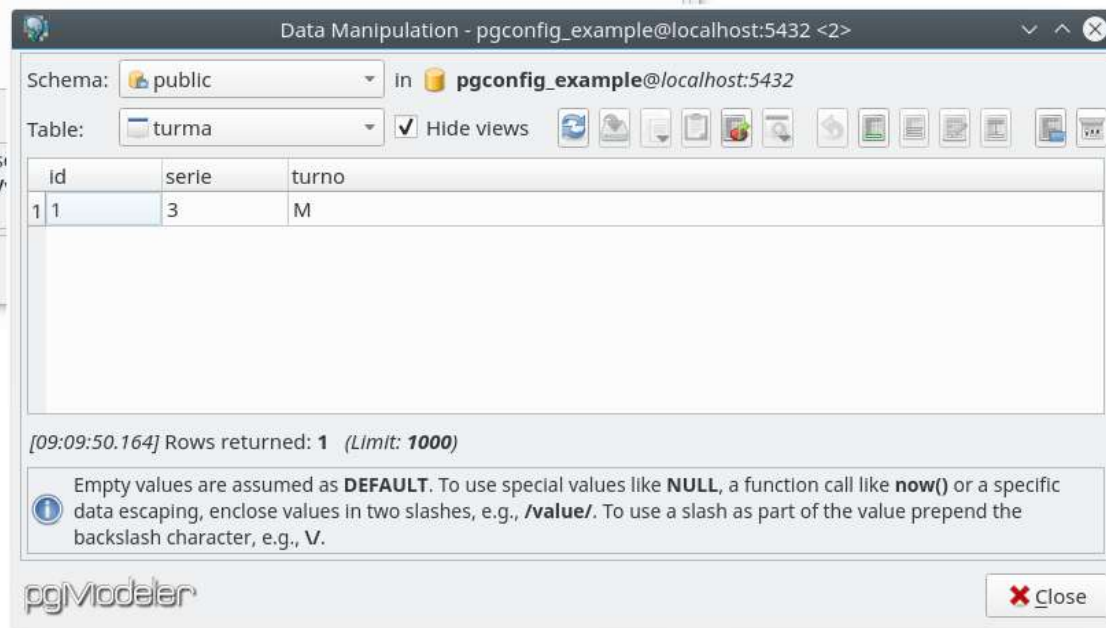
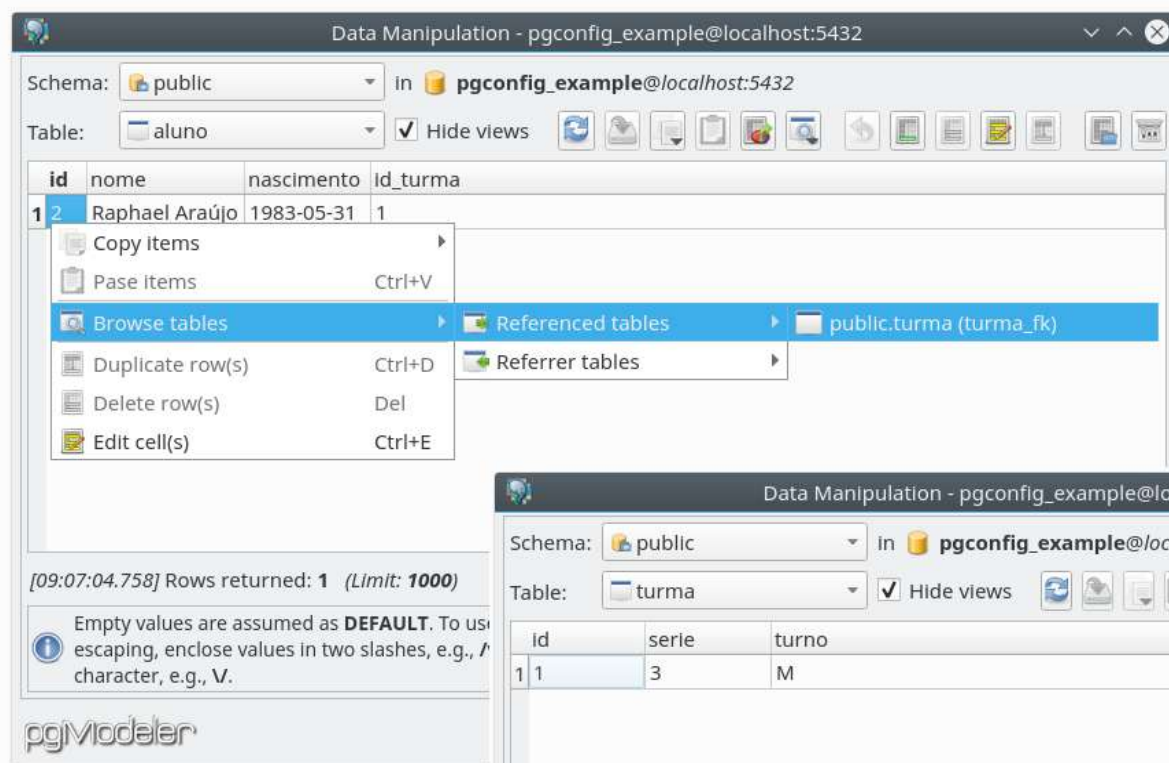
[17:32:16.931] Rows returned: 3 (Limit: 1000)

Empty values are assumed as **DEFAULT**. To use special values like **NULL**, a function call like **now()** or a specific data escaping, enclose values in two slashes, e.g., **/value/**. To use a slash as part of the value prepend the backslash character, e.g., **V.**

pgModeler Close

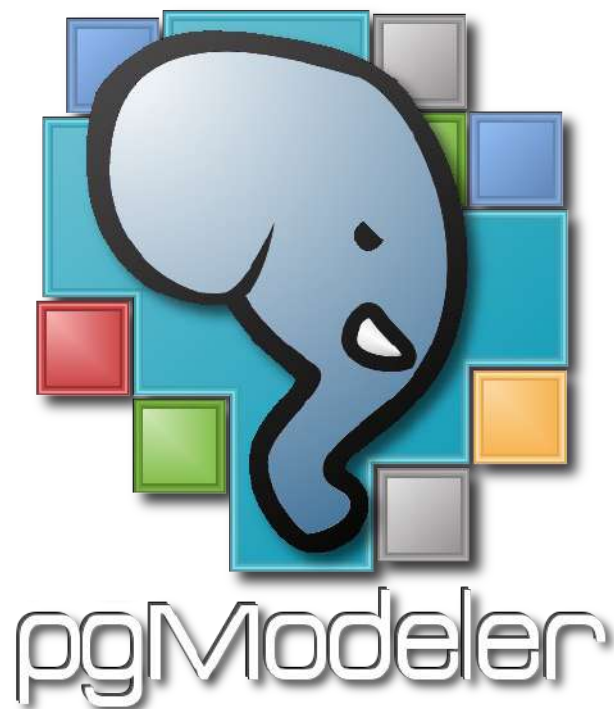
Manipulação de dados

- Navegação entre registros:



E no futuro?

- Melhorias na interface e usabilidade;
- Suporte ao particionamento de tabelas;
- Suporte a criação de objetos através formulários no módulo de administração;
- Engenharia reversa de bancos de dados MySQL / MariaDB;
- ...e muitas outras!



Muito obrigado!

<https://pgmodeler.io>

raphael@pgmodeler.io